

Đã bắt đầu vào lúc	Thứ hai, 21 Tháng mười một 2022, 2:53 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ bảy, 3 Tháng mười hai 2022, 9:49 AM
Thời gian thực hiện	11 ngày 18 giờ
Điểm	5,00/5,00
Điểm	10,00 của 10,00 (100%)



Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

Implement Depth-first search

```
Adjacency *BFS(int v);
```

where Adjacency is a structure to store list of number.

```
#include <iostream>
#include <list>
using namespace std;

class Adjacency
{
private:
    list<int> adjList;
    int size;
public:
    Adjacency() {}
    Adjacency(int V) {}
    void push(int data)
    {
        adjList.push_back(data);
        size++;
    }
    void print()
    {
        for (auto const &i : adjList)
            cout << " -> " << i;
    }
    void printArray()
    {
        for (auto const &i : adjList)
            cout << i << " ";
    }
    int getSize() { return adjList.size(); }
    int getElement(int idx)
    {
        auto it = adjList.begin();
        advance(it, idx);
        return *it;
    }
};
```

And Graph is a structure to store a graph (see in your answer box)

For example:

Test	Result
<pre>int V = 6; int visited = 0; Graph g(V); Adjacency* arr = new Adjacency(V); int edge[][2] = {{0,1},{0,2},{1,3},{1,4},{2,4},{3,4},{3,5},{4,5}}; for(int i = 0; i < 8; i++) { g.addEdge(edge[i][0], edge[i][1]); } arr = g.BFS(visited); arr->printArray(); delete arr;</pre>	0 1 2 3 4 5
<pre>int V = 6; int visited = 2; Graph g(V); Adjacency* arr = new Adjacency(V); int edge[][2] = {{0,1},{0,2},{1,3},{1,4},{2,4},{3,4},{3,5},{4,5}}; for(int i = 0; i < 8; i++) { g.addEdge(edge[i][0], edge[i][1]); } arr = g.BFS(visited); arr->printArray(); delete arr;</pre>	2 0 4 1 3 5

Answer: (penalty regime: 0, 0, 5, 10, ... %)

Reset answer

```
1 #include <queue>
2 class Graph
3 {
4 private:
5     int V;
6     Adjacency *adj;
7
8 public:
9     Graph(int V)
10    {
11        this->V = V;
12        adj = new Adjacency[V];
13    }
14
15    void addEdge(int v, int w)
16    {
17        adj[v].push(w);
18        adj[w].push(v);
19    }
20
21    void printGraph()
22    {
23        for (int v = 0; v < V; ++v)
24        {
25            cout << "\nAdjacency list of vertex " << v << "\nhead ";
26            adj[v].print();
27        }
28    }
29
30    Adjacency *BFS(int v)
31    {
```

```

32 Adjacency* ans = new Adjacency();
33 queue<int> q;
34 int* color = new int[V];
35 for(int i = 0; i < V; i++) color[i] = -1;
36 q.push(v);
37 color[v] = 1;
38 ans->push(v);
39 while(!q.empty()){
40     int u = q.front();
41     q.pop();
42     for(int i = 0; i < adj[u].getSize(); i++){
43         if(color[adj[u].getElement(i)] == -1){
44             q.push(adj[u].getElement(i));
45             color[adj[u].getElement(i)] = 1;
46             ans->push(adj[u].getElement(i));
47         }
48     }
49 }
50 }

```

	Test	Expected	Got	
✓	<pre> int V = 6; int visited = 0; Graph g(V); Adjacency* arr = new Adjacency(V); int edge[][2] = {{0,1},{0,2},{1,3},{1,4},{2,4},{3,4},{3,5},{4,5}}; for(int i = 0; i < 8; i++) { g.addEdge(edge[i][0], edge[i][1]); } arr = g.BFS(visited); arr->printArray(); delete arr; </pre>	0 1 2 3 4 5	0 1 2 3 4 5	✓
✓	<pre> int V = 6; int visited = 2; Graph g(V); Adjacency* arr = new Adjacency(V); int edge[][2] = {{0,1},{0,2},{1,3},{1,4},{2,4},{3,4},{3,5},{4,5}}; for(int i = 0; i < 8; i++) { g.addEdge(edge[i][0], edge[i][1]); } arr = g.BFS(visited); arr->printArray(); delete arr; </pre>	2 0 4 1 3 5	2 0 4 1 3 5	✓

	Test	Expected	Got	
✓	<pre>int V = 8, visited = 5; Graph g(V); Adjacency *arr; int edge[][2] = {{0,1}, {0,2}, {0,3}, {0,4}, {1,2}, {2,5}, {2,6}, {4,6}, {6,7}}; for(int i = 0; i < 9; i++) { \tg.addEdge(edge[i][0], edge[i][1]); } // g.printGraph(); // cout << endl; arr = g.BFS(visited); arr->printArray(); delete arr;</pre>	5 2 0 1 6 3 4 7	5 2 0 1 6 3 4 7	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.



Câu hỏi 2

Chính xác

Điểm 1,00 của 1,00

Implement Depth-first search

```
Adjacency *DFS(int v);
```

where Adjacency is a structure to store list of number.

```
#include <iostream>
#include <list>
using namespace std;

class Adjacency
{
private:
    list<int> adjList;
    int size;
public:
    Adjacency() {}
    Adjacency(int V) {}
    void push(int data)
    {
        adjList.push_back(data);
        size++;
    }
    void print()
    {
        for (auto const &i : adjList)
            cout << " -> " << i;
    }
    void printArray()
    {
        for (auto const &i : adjList)
            cout << i << " ";
    }
    int getSize() { return adjList.size(); }
    int getElement(int idx)
    {
        auto it = adjList.begin();
        advance(it, idx);
        return *it;
    }
};
```

And Graph is a structure to store a graph (see in your answer box)

For example:

Test	Result
<pre> int V = 8, visited = 0; Graph g(V); Adjacency *arr; int edge[][2] = {{0,1}, {0,2}, {0,3}, {0,4}, {1,2}, {2,5}, {2,6}, {4,6}, {6,7}}; for(int i = 0; i < 9; i++) { g.addEdge(edge[i][0], edge[i][1]); } // g.printGraph(); // cout << endl; arr = g.DFS(visited); arr->printArray(); delete arr; </pre>	0 1 2 5 6 4 7 3

Answer: (penalty regime: 0, 0, 5, ... %)

Reset answer

```

1 class Graph
2 {
3 private:
4     int V;
5     Adjacency *adj;
6
7 public:
8     Graph(int V)
9     {
10         this->V = V;
11         adj = new Adjacency[V];
12     }
13
14     void addEdge(int v, int w)
15     {
16         adj[v].push(w);
17         adj[w].push(v);
18     }
19
20     void printGraph()
21     {
22         for (int v = 0; v < V; ++v)
23         {
24             cout << "\nAdjacency list of vertex " << v << "\nhead ";
25             adj[v].print();
26         }
27     }
28
29     void DFSVisited(Adjacency* ans, int v, int* color){
30         ans->push(v);
31         int size = adj[v].getSize();
32         color[v]++;
33         for(int i = 0; i < size; i++){
34             if(color[adj[v].getElement(i)] == -1) DFSVisited(ans,adj[v].getElement(i),color);
35         }
36         color[v]++;
37     }
38
39     Adjacency *DFS(int v)
40     {
41         int* color = new int[V];
42         for(int i = 0; i < V; i++) color[i] = -1;
43         Adjacency* ans = new Adjacency();
44         DFSVisited(ans,v,color);
45         return ans;
46     }
47 };

```



	Test	Expected	Got	
✓	<pre>int V = 8, visited = 0; Graph g(V); Adjacency *arr; int edge[][2] = {{0,1}, {0,2}, {0,3}, {0,4}, {1,2}, {2,5}, {2,6}, {4,6}, {6,7}}; for(int i = 0; i < 9; i++) { \tg.addEdge(edge[i][0], edge[i][1]); } // g.printGraph(); // cout << endl; arr = g.DFS(visited); arr->printArray(); delete arr;</pre>	0 1 2 5 6 4 7 3	0 1 2 5 6 4 7 3	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

BACHKHOACNCP.COM

TÀI LIỆU SƯU TẬP

BỞI HCMUT-CNCP

Câu hỏi 3

Chính xác

Điểm 1,00 của 1,00

The relationship between a group of people is represented by an adjacency-list `friends`. If `friends[u]` contains `v`, `u` and `v` are friends. Friendship is a two-way relationship. Two people are in a friend group as long as there is some path of mutual friends connecting them.

Request: Implement function:

```
int numberOfFriendGroups(vector<vector<int>>& friends);
```

Where `friends` is the adjacency-list representing the friendship (this list has between 0 and 1000 lists). This function returns the number of friend groups.

Example:

Given a adjacency-list: `[[1], [0, 2], [1], [4], [3], []]`

There are 3 friend groups: `[0, 1, 2], [3, 4], [5]`

Note:

In this exercise, the libraries `iostream`, `string`, `cstring`, `climits`, `utility`, `vector`, `list`, `stack`, `queue`, `map`, `unordered_map`, `set`, `unordered_set`, `functional`, `algorithm` have been included and `namespace std` is used. You can write helper functions and class. Importing other libraries is allowed, but not encouraged.

For example:

Test	Result
<pre>vector<vector<int>> graph { {1}, {0, 2}, {1}, {4}, {3}, {} }; cout << numberOfFriendGroups(graph);</pre>	3

Answer: (penalty regime: 0, 0, 0, 5, 10, ... %)

Reset answer

```
1 int BFSSearch(vector<vector<int>>& friends) {
2     queue<int> q;
3     if(int(friends.size()) == 0) return 0;
4     int* color = new int[int(friends.size())];
5     for (int i = 0; i < int(friends.size()); i++) {
6         color[i] = -1;
7     }
8
9     int count = 0;
10    bool isDone = false;
11    int start = 0;
12    while (!isDone) {
13        q.push(start);
14        color[start] = 1;
15        isDone = true;
16        while (!q.empty()) {
17            int u = q.front();
18            q.pop();
19            int size = int(friends[u].size());
20            for (int i = 0; i < size; i++) {
21                if (color[friends[u][i]] == -1) {
22                    q.push(friends[u][i]);
```

	Test	Expected	Got	
✓	<pre>vector<vector<int>> graph { \t{1}, \t{0, 2}, \t{1}, \t{4}, \t{3}, \t{} }; cout << numberOfFriendGroups(graph);</pre>	3	3	✓
✓	<pre>vector<vector<int>> graph { }; cout << numberOfFriendGroups(graph);</pre>	0	0	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.



Câu hỏi 4

Chính xác

Điểm 1,00 của 1,00

Implement function to detect a cyclic in Graph

```
bool isCyclic();
```

Graph structure in this lab is slightly different from previous labs.

```
#include<iostream>
#include <list>
using namespace std;

class DirectedGraph
{
    int V;
    list<int> *adj;
    bool isCyclicUtil(int v, bool visited[], bool *rs);
public:
    DirectedGraph(){
        V = 0;
        adj = NULL;
    }
    DirectedGraph(int V)
    {
        this->V = V;
        adj = new list<int>[V];
    }
    void addEdge(int v, int w)
    {
        adj[v].push_back(w);
    }
    bool isCyclic();
};
```



For example:

Test	Result
<pre>DirectedGraph g(8); int edege[][2] = {{0,6}, {1,2}, {1,4}, {1,6}, {3,0}, {3,4}, {5,1}, {7,0}, {7,1}}; for(int i = 0; i < 9; i++) g.addEdge(edege[i][0], edege[i][1]); if(g.isCyclic()) cout << "Graph contains cycle"; else cout << "Graph doesn't contain cycle";</pre>	Graph doesn't contain cycle

Answer: (penalty regime: 0, 0, 5, ... %)

Reset answer

```
1 #include<iostream>
2 #include <list>
3 #include <bits/stdc++.h>
4 using namespace std;
5
6 class DirectedGraph
7 {
```

```
8   int V;
9   list<int> *adj;
10  bool isCyclicUtil(int v, bool visited[], bool *rs);
11  public:
12  DirectedGraph(){
13      V = 0;
14      adj = NULL;
15  }
16  DirectedGraph(int V)
17  {
18      this->V = V;
19      adj = new list<int>[V];
20  }
21  void addEdge(int v, int w)
22  {
23      adj[v].push_back(w);
24  }
25  bool DFSSearch(int u, int* color){
26      color[u]++;
27      int size = adj[u].size();
28      for(int i = 0; i < size; i++){
29          auto it = adj[u].begin();
30          advance(it,i);
31          int v = *it;
32          if(color[v] == -1){
33              if(DFSSearch(v,color)) ;
34              else return false;
35          }
36          if(color[v] == 0){
37              return false;
38          }
39      }
40      color[u]++;
41      return true;
42  }
43  bool isCyclic()
44  {
45      int* color = new int[V];
46      //int count = 0;
47      for(int i = 0; i < V; i++) color[i] = -1;
48      for(int i = 0; i < V; i++){
49          if(color[i] == -1){
50              if(DFSSearch(i,color));
51              else{
52                  return true;
53              }
54          }
55      }
56      return false;
57  }
58  };
```



	Test	Expected	Got	
✓	<pre> DirectedGraph g(8); int edge[][2] = {{0,6}, {1,2}, {1,4}, {1,6}, {3,0}, {3,4}, {5,1}, {7,0}, {7,1}}; for(int i = 0; i < 9; i++) \tg.addEdge(edge[i][0], edge[i][1]); if(g.isCyclic()) \tcout << "Graph contains cycle"; else \tcout << "Graph doesn't contain cycle"; </pre>	Graph doesn't contain cycle	Graph doesn't contain cycle	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.



Câu hỏi 5

Chính xác

Điểm 1,00 của 1,00

Implement **topologicalSort** function on a graph. (Ref [here](#))

```
void topologicalSort();
```

where Adjacency is a structure to store list of number. Note that, the vertex index starts from 0. **To match the given answer, please always traverse from 0 when performing the sorting.**

```
#include <iostream>
#include <list>
using namespace std;

class Adjacency
{
private:
    list<int> adjList;
    int size;
public:
    Adjacency() {}
    Adjacency(int V) {}
    void push(int data)
    {
        adjList.push_back(data);
        size++;
    }
    void print()
    {
        for (auto const &i : adjList)
            cout << " -> " << i;
    }
    void printArray()
    {
        for (auto const &i : adjList)
            cout << i << " ";
    }
    int getSize() { return adjList.size(); }
    int getElement(int idx)
    {
        auto it = adjList.begin();
        advance(it, idx);
        return *it;
    }
};
```

And Graph is a structure to store a graph (see in your answer box). You could write one or more helping functions.

For example:

Test	Result
<pre> Graph g(6); g.addEdge(5, 2); g.addEdge(5, 0); g.addEdge(4, 0); g.addEdge(4, 1); g.addEdge(2, 3); g.addEdge(3, 1); g.topologicalSort(); </pre>	5 4 2 3 1 0

Answer: (penalty regime: 0, 0, 5, 10, ... %)

Reset answer

```

1 #include <vector>
2 class Graph {
3
4     int V;
5     Adjacency* adj;
6
7 public:
8     Graph(int V){
9         this->V = V;
10        adj = new Adjacency[V];
11    }
12    void addEdge(int v, int w){
13        adj[v].push(w);
14    }
15
16    bool DFS(int u, int* color, vector<int>& vec){
17
18        color[u]++;
19        int size = adj[u].getSize();
20        for(int i = 0; i < size; i++){
21            int v = adj[u].getElement(i);
22            if(color[v] == -1){
23                if(DFS(v,color,vec));
24                else return false;
25            }
26            if(color[v] == 0) return false;
27        }
28        vec.insert(vec.begin(),u);
29        color[u]++;
30        return true;
31    }
32    void printVec(vector<int>& vec){
33        auto it = vec.begin();
34        if(it != vec.end()) cout << *it;
35        else return;
36        it++;
37        for(it = it; it != vec.end(); it++){
38            cout << " " <<*it;
39        }
40    }
41    void topologicalSort(){
42        int* color = new int[V];
43        vector<int> vec;
44        for(int i = 0; i < V; i++) color[i] = -1;
45        for(int u = 0; u < V; u++){
46            if(color[u] == -1){
47                if(DFS(u,color,vec)){
48
49                }
50            }
51        }
52    }

```

	Test	Expected	Got	
✓	<pre> Graph g(6); g.addEdge(5, 2); g.addEdge(5, 0); g.addEdge(4, 0); g.addEdge(4, 1); g.addEdge(2, 3); g.addEdge(3, 1); g.topologicalSort(); </pre>	5 4 2 3 1 0	5 4 2 3 1 0	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

BÁCH KHOA E-LEARNING



WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

📍 268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

☎ (028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

✉ elearning@hcmut.edu.vn



Copyright 2007-2022 BKEL - Phát triển dựa trên Moodle